PCIe-BaseLab

Tarjeta de evaluación para el bus PCI Express

Versión 1.02
Junio 2015

Indicaciones acerca del volumen de suministro, la instalación, las propiedades del hardware y sus posibilidades de ampliación, así como acerca de la programación sobre la base de las funciones API- Funciones

Índice

1. Introducción ... 3
   - Características generales ... 3
   - Volumen de suministro ... 3

2. Instalación ... 4
   - Software ... 4
   - Hardware ... 4
   - Asignación del driver de Windows ... 4

3. Propiedades del hardware y posibilidades de ampliación ... 5
   - Controlador PCIe Endpoint Controller PEX8311 ... 5
   - Interfaz de bus PCI Express ... 5
   - Interfaz de bus local ... 5
   - Configuración serial EEPROM ... 6
   - Dispositivo lógico programable(PLD), ispXPLD5768 ... 6
   - “In-System Programmable” (ISP) ... 6
   - Pin Header Block (User Interface) ... 6

4. Las aplicaciones de ejemplo ... 7
   - RAM síncrona ... 7
   - FIFO (“primero en entrar, primero en salir”) ... 7
   - Registro de entradas/salidas de 16 bits ... 7

5. La interfaz de programación ... 8
   - Utilización de los archivos ... 8
   - Carga de los drivers ... 8
   - Formato de las funciones, código de error ... 8
   - Configuraciones de las memorias ... 8
   - Métodos de acceso a la memoria ... 9

6. API - Funciones - Referencia ... 10
   - Consulta del número de PCIe-BaseLab-tarjetas activas ... 11
   - Apertura de una PCIe-BaseLab-tarjeta ... 11
   - Cierre de una PCIe-BaseLab-tarjeta ... 11
   - Consulta de la versión del driver de Windows ... 12
   - Consulta de informaciones sobre la ranura de PCIe-extensión ... 12
   - Reinicio de una PCIe-BaseLab-tarjeta ... 12
   - Lectura de los registros existentes en la PCI-zona de configuración ... 13
   - Escritura de los registros existentes en la PCI-zona de configuración ... 13
   - Lectura de los registros del PEX8311 controlador ... 13
   - Escritura de los registros del PEX8311 controlador ... 13
   - Lectura de datos de la EEPROM ... 14
   - Escritura de datos en la EEPROM ... 14
   - Lectura de la zona de memoria ... 14
   - Escritura de la zona de memoria ... 14
   - “Intercalación” de una zona de memoria en el espacio de direcciones de la aplicación ... 15
   - “Enmascaramiento” de una zona de memoria fuera del espacio de direcciones de la aplicación ... 15
   - Consulta de la dirección física de las “Regiones de usuario” ... 15
   - Adjudicación de una zona de memoria relacionada ... 16
   - Activación de la zona de memoria relacionada ... 16

7. Los programas de ejemplo (C++) ... 17

8. El programa de monitorización PPLABMON.EXE ... 18
   - Procesamiento del registro de la EEPROM ... 19
   - Lectura y escritura en la memoria RAM ... 20
   - Fijación y consulta de los pines del 16-Bits-I/O-registro ... 21

Anexo ... 22

1. Introducción

Características generales

PCIe-BaseLab es un medio imprescindible para el desarrollo de tarjetas adicionales para ordenadores personales y otros sistemas informáticos que están equipados con el bus PCI Express (PCIe). Esta tarjeta permite realizar un test rápido y sencillo de los circuitos electrónicos recién desarrollados en el bus PCI Express.

PCIe-BaseLab funciona con el controlador universal PCI Express Endpoint Controller *PEX8311 (PLX Technology, Inc.)*, con su equipo periférico y un dispositivo lógico programable (PLD, Lattice Semiconductor) de alto rendimiento. Con el *PEX8311* controlador, *PCIe-BaseLab* es compatible con los requisitos de la PCI Express especificación, Revision 1.0. La tarjeta está completamente equipada y probada, de modo que se suministra lista para su aplicación.

El diseñador de los circuitos puede montar su hardware directamente en la placa para circuito experimental de la tarjeta secundaria y comenzar los test. La ocupación con los juegos de señales y las características técnicas del sistema de bus puede reducirse al mínimo.

El dispositivo lógico programable contiene diversas aplicaciones de ejemplo preinstaladas:

- una RAM sincrónica,
- una FIFO ("primero en entrar, primero en salir")
- un registro de entradas/salidas de 16 bits.

Se suministran los códigos fuente de las aplicaciones a modo de ejemplo, y pueden ser modificados por el usuario.

Una característica especialmente ventajosa para los diseñadores de hardware y técnicos de medición: Todos los pines de conexión local del controlador están vinculados de antemano al dispositivo lógico programable (PLD), conducido sobre regletas de patas y, a partir de allí, resultan cómodamente accesibles. 106 de los 193 pines de entrada/salida del dispositivo lógico programable están dispuestos igualmente sobre regletas de patas y completamente disponibles.
Para *PCIe-BaseLab* existen tarjetas secundarias en diferentes variantes. En este momento, con una placa para circuito experimental de 0,1 pulgadas, más tarde también con espacios para dispositivos modulares de almacenamiento (SMD) o con una forma específica según cada cliente. Esto último deja abierta la posibilidad de aplicar *PCIe-BaseLab* como módulo junto con hardware específico de cada aplicación en las series pequeñas y medianas.

La aplicación de *PCIe-BaseLab* no genera costes de licencia para un núcleo, ni tampoco es necesario ser miembro del PCI-SIG. Los sub-proveedores y las IDs de subsistema destinados a la identificación unívoca del hardware propio pueden obtenerse gratuitamente del fabricante del controlador PCI Express.

La documentación de *PCIe-BaseLab* se completa con los diagramas de circuito, los esquemas de equipamiento y conexiones, así como con códigos fuente.

Volumen de suministro

El volumen de suministro del producto *PCIe-BaseLab* incluye los siguientes componentes:

- *Tarjeta de evaluación para el bus PCI Express* con tarjeta secundaria incorporada (trama de terminales de 0,1 pulgadas),
- *Cable adaptador* para el abastecimiento de corriente de la tarjeta secundaria por medio de un conector de abastecimiento de corriente SATA,
- *CD-ROM*: drivers de Windows, programa de monitorización, aplicaciones de ejemplo incl. texto fuente y archivos de programación (Header, Lib, DLL), así como esquemas de conexiones, hojas de datos, códigos fuente y el presente manual técnico.

2. Instalación

Software

El *PCIe-BaseLab* software puede utilizarse con Windows2000, WindowsXP, Windows Vista y Windows7. El driver versiona de Windows existirá de 32-Bit plataformas y plataformas de 64 bits.

ATENCIÓN: I para instalar el software debe estar en posesión de los derechos de administrador!

Para instalar el software, seleccione uno de los siguientes modos de procedimiento (dependiendo de la variante suministrada):

- Copie toda la estructura de directorios del CD en el directorio de destino de su PC.
- Descomprima los archivos iniciando el programa de autodescompresión 'pplab_v1005.exe'.
- Inicie la instalación ejecutando el 'Setup.exe'.

En el directorio del software encontrará los siguientes subdirectorios:

- bin: los controladores -API-DLL 'PPLABAPI.DLL' (→ Apartado 5.), así como los datos binarios de los programas de ejemplo (→ Apartado 8.),
- dev: los dos archivos de encabezado C Header 'PPLABAPI.H' y 'PPLABERR.H' (→ Apartado 5.),
- doc: el manual,
- drv: el driver de Windows'PPLAB_E.SYS', así como los 2 archivos INF correspondientes (→ siguiente Apartado),
- lib: la biblioteca de importación 'PPLABAPI.LIB' (→ Apartado 5.),
- mon: el programa de monitorización 'PPLABMON.EXE', así como sus archivos correspondientes (→ Apartado 7.),
- smp: los archivos de códigos fuente de los programas de ejemplo (→ Apartado 8.).

Recomendación: Tiene sentido instalar el software antes de incorporar la tarjeta *PCIe-BaseLab* al PC. De ese modo, la asignación del driver de Windows se produce inmediatamente tras reiniciar el ordenador.

Hardware

La instalación del hardware se realiza introduciendo la tarjeta en la ranura PCI Express x1, x4, x8 o x16 de un PC o de otro sistema informático con ranuras de extensión para PCI Express, teniendo en cuenta las leyes de seguridad de descarga electroestática y la protección contra el contacto con componentes que puedan encontrarse sometidos a una tensión eléctrica peligrosa.

Antes de efectuar la instalación/desinstalación, el ordenador debe apagarse y desconectarse de la red.

Tras encender el ordenador, el LED (LED1) integrado en la tarjeta muestra la realización de la conexión de enlace físico de la PCIe-interfaz, y su iluminación sirve a modo de primer control de una instalación correcta del hardware.

El funcionamiento de *PCIe-BaseLab* también es posible a través de un PCIe-Bus-extensor. Esto permite la introducción y extracción de la tarjeta con el PC en funcionamiento (Hot-Swapping) sin que se produzca una pérdida de los datos de configuración. (*PCIe-Bus-Extender* con *PCFaceSwitch-Software*, *HK Meßsysteme GmbH*)

Asignación del driver de Windows

Tras el reinicio del ordenador, el sistema operativo encuentra el hardware nuevo y requiere la asignación de un driver apropiado. En este sentido, hay que tener en cuenta lo siguiente: El PLX-controlador PEX8311 utilizado está formado por dos componentes internos a saber:

- por una parte, el "puente PCI-PCI" (PCI8111) y
- por la otra, el "PCI-dispositivo" (PCI9056).

ATENCIÓN: I Ambos componentes deben asignarse a un driver apropiado!

Para realizar una primera asignación de los drivers, no utilice la recomendada "instalación automática", remita al "Asistente de hardware" existente en el diálogo correspondiente al subdirectorio "drv" existente en el directorio del *PCIe-BaseLab*-software. Allí, los dos INF-archivos

- PLXBRIDGE.INF y
- PPLAB_E.INF

se ocupan de la asignación de los drivers correctos. Éstos son

- PCI.SYS – el driver estándar de Microsoft para el bus PCI, aquí en forma de driver para el "puente PCI-PCI",
- PPLAB_E.SYS – el driver especial de Windows para el "PCI-dispositivo" *PCIe-BaseLAb.*

En caso de que el "PCI-dispositivo" haya sido asignado a otro driver de Windows (por ejemplo, el driver PLX genérico), asigne a posteriori al "PCI-dispositivo" el driver correcto. Para ello, remítase a Administrador de dispositivos – Actualizar drivers – Directorio "drv", en "PPLAB_E.INF". Como resultado de una asignación de drivers exitosa, encontrará en el Administrador de dispositivos la clase de dispositivo "PCIe-BaseLab" y allí los dispositivos "PCIe-BaseLab" instalados.

3. Propiedades del hardware y posibilidades de ampliación

El diagrama funcional (ver Anexo) sirve de descripción introductoria. Desde el punto de vista funcional, el hardware se divide en cinco partes:

- PCIe Endpoint Controller PEX8311,
- Serial Configuration EEPROM,
- Programmable Logic Device (PLD), ispXPLD5768,
- 'In-System Programmable' Interface (ISP),
- Pin Header Block (User Interface)

Controlador PCIe Endpoint Controller PEX8311

El *PEX8311* es un PCIe Endpoint controlador universal para el bus PCI Express serial, equipado con un enlace x1. Sirve como puente entre el PCIe-bus y los circuitos específicos de cada usuario existentes en el bus local. El *PEX8311* procesa todas las señales y mecanismos de acceso típicos existentes en el PCIe-bus. Los traslada a una interfaz de control, direcciones y datos, a la que se pueden conectar memorias y unidades de entrada/salida específicas de cada usuario. Para ello, dispone de los interfaces, designadas de la manera siguiente:

- PCIe Bus Interface y
- Local Bus Interface

Las interfaces del PEX8311 controlador poseen diferente significación para el usuario de la *PCIe-BaseLab*-tarjeta, y las describiremos a continuación.

Interfaz de bus PCI Express

Esta *PCIe-Bus-interfaz* sirve para acoplar el controlador al PCI Express bus. Ya está completamente cableada en el circuito impreso, y el usuario no debe dedicar ningún esfuerzo en ese sentido.

Interfaz de bus local

Esta *interfaz de bus local* es importante para el usuario, dado que aquí conectará su aplicación de circuito. Está dispuesta de manera universal y permite el funcionamiento del equipo periférico de hardware con amplitudes del bus de datos de 8, 16 o 32 bits, así como una amplitud del bus de direcciones de un máximo de 32 bits.

Los componentes específicos de cada usuario controlan el intercambio de datos a través de la interfaz de manera funcional con la ayuda de señales clásicas.

El equipo periférico conectado puede estar formado por un sistema de microprocesadores o (en el caso más sencillo) estar configurado a modo de latch de datos.

La arquitectura del controlador permite también la incorporación de memorias. Es posible dirigir hasta 4 GB por cada región, y son posibles dos regiones locales.

El *PEX8311* dispone de un conjunto de registros interno a fin de almacenar los datos de inicialización, realizar el ajuste, la activación y la desactivación de los modos de funcionamiento o intercambiar datos. Es posible acceder al conjunto de registros tanto a través de la interfaz como a través de la interfaz de bus. Así, las funciones del PCI-controlador resultan transparentes y utilizables para ambos lados.

Para la transferencia rápida de datos o la utilización de la Host-CPU (Bus Master Transfer Mode), hay previstos dos DMA-canales independientes, cuyas direcciones de inicio y contadores de transferencia también pueden ajustarse a través del registro.

Para la ampliación del sistema básico de entrada y salida (BIOS) es posible conectar al *PEX8311* memorias de sólo lectura (ROM) con una interfaz paralela.

El controlador *PEX8311* también permite la generación específica según el usuario de peticiones de interrupción, que pueden proceder tanto de la parte local como de la conexión PCI.

Un manual técnico específico editado y supervisado por el fabricante del controlador recoge la descripción técnica completa del *PEX8311*. Dicho manual puede descargarse de las páginas web*PLX Technology, Inc.* o extraerse del CD-ROM suministrado.

Atención: No podemos garantizar la vigencia de las hojas de datos de otros fabricantes suministradas con este producto. Éstas deben utilizarse exclusivamente como un primer nivel de información. No obstante, siempre es recomendable hacerse con las hojas de datos y manuales más actualizados del respectivo fabricante, y utilizarlos como fundamento en la propia actividad a realizar con los componentes. Puede consultar las correspondientes direcciones en el Anexo al presente manual.

Configuración serial EEPROM

La *PCIe-BaseLab* funciona con una memoria EEPROM serial de 2kbs de tamaño, que en una versión está instalada de manera encajable en el circuito impreso. Contiene datos de configuración obligatorios que inicializan el controlador especialmente para la aplicación de la *PCIe-BaseLab*. La memoria EEPROM puede editarse y sobrescribirse. Un editor práctico para el código de la memoria EEPROM serial está incluido en el paquete de software suministrado, en forma del programa *PPLABMON.EXE*.

Dispositivo lógico programable (PLD), ispXPLD5768

El dispositivo lógico programable utilizado en la *PCIe-BaseLab* tarjeta pertenece a la familia *ispXPLD 5000MX Lattice Semiconductor*. Esta PLD serie se caracteriza por un gran rendimiento y flexibilidad. Además de la lógica exhaustiva, también es posible implementar memorias single y RAM en los bloques que contiene. Las entradas y las salidas pueden funcionar con distintos estándares de interfaz. Los sistemas integrados permiten una gestión sencilla de los impulsos. La serie dispone de una memoria integrada para la configuración que hace que el dispositivo lógico programable esté disponible inmediatamente tras la conexión de la tensión de alimentación. El dispositivo lógico programable utilizado en la tarjeta posee 768 macrocélulas y una RAM configurable a un máximo de 384 kbits, y es posible aplicarle una frecuencia de impulsos de sistema de un máximo de 250 MHz. Sus salidas proporcionan un centro de nivel de 3,3 V, y las entradas son compatibles con 5 V. La programación del dispositivo lógico programable puede realizarse a través de la interfaz integrada en la tarjeta *PCIe-BaseLab*, y no es necesario un dispositivo de programación externo. El lector podrá encontrar más información sobre el dispositivo lógico programable en las páginas web de la empresa *Lattice Semiconductor*.

"In-System Programmable" Interface (ISP)

La ISP interfaz (X1, JEDEC-Support) permite la descarga de PLD-firmware sin necesidad de procesar el dispositivo lógico programable en un dispositivo de programación externo. Es posible que permanezca integrado en el circuito ('in system programmable', ISP). El software *ispVMSystem* necesario para ello puede descargarse de manera gratuita de las páginas web de la empresa *Lattice Semiconductor*. Cable de programación bien encargarse a modo de versión en paralelo o en USB en su distribuidor local de componentes *Lattice*.

Pin Header Block (User Interface)

El *pin header block* (bloque terminal de pines) está compuesto de conectores de patas en dos hileras en cuadrícula de 2 mm (2 dos piezas con 2x50 pines cada una, 1 pieza con 2x20 pines). Éstos están dispuestos en los bordes exteriores de la tarjeta, y forman la interfaz hacia el hardware específico del usuario.

Dicha interfaz permite el acceso a las conducciones locales de direcciones, datos y control del controlador *PEX8311*, así como a 106 de los 193 pines de entrada/salida del dispositivo lógico programable *ispXPLD5768* instalado. Algunas otras conexiones permiten la recepción de señales de impulso o su alimentación externa. Los pines de masa están dispuestos entre sí guardando distancias equivalentes.

La estructura de la tarjeta secundaria suministrada está diseñada de modo que pueda insertarse en los conectores de patas de la *PCIe-BaseLab* tarjeta y ser atornillada a ellos. Para ello, se transmiten todos los pines de la tarjeta secundaria y quedan allí preparados para la conexión de hardware específico del usuario. Atención: Los suministros de corriente y las conexiones de masa no se enlazarán automáticamente con las capas de suministro de la tarjeta secundaria. Para ello es necesario un cable adicional (igualmente incluido el suministro), que puede unirse a elección o bien a una clavija de suministro de corriente del ordenador huésped o bien a las conexiones soldadas preparadas en la tarjeta. Todas las alimentaciones de tensión (con excepción de la conexión de masa) están realizadas en la tarjeta secundaria a modo de fusibles recambiables, cuya función es proteger la electrónica del usuario y el módulo de suministro de corriente del ordenador huésped contra excesos de corriente. Los LEDs de color señalizan la presencia de las tensiones de suministro en la tarjeta secundaria.

La separación entre *PCIe-BaseLab* tarjeta y tarjeta secundaria es posible, si bien debe realizarse de manera muy cuidadosa y con la ayuda de una herramienta, a causa de la potente acción de la fuerza de los conectores (por favor, levantar de manera uniforme y en paralelo utilizando un destornillador apropiado).

4. Las aplicaciones de ejemplo

Las aplicaciones de ejemplo están incluidas en el dispositivo lógico programable exclusivamente a modo de circuitos lógicos o "glue logic". Deben servir para mostrar a modo de ejemplo posibles aplicaciones de hardware con las que puede equiparse la *PCIe-BaseLab* sin necesidad de instalar componentes adicionales en la tarjeta secundaria.

El usuario puede adaptar a sus necesidades las aplicaciones de ejemplo preinstaladas dentro del marco de sus posibilidades técnicas y/o implementar también su propia lógica en el dispositivo lógico programable.

Los códigos fuente de las aplicaciones de ejemplo van incluidos en el paquete de producto. Para el procesamiento y la compilación se necesita el software correspondiente, que puede descargarse gratuitamente en las páginas web de la empresa *Lattice Semiconductor*.

El programa de monitorización suministrado con este producto permite la comunicación con las aplicaciones de ejemplo con el fin de realizar un test. Más información al respecto: Apartado 8.

A continuación, el lector recibirá otras informaciones acerca de la arquitectura técnica de hardware de las aplicaciones de ejemplo preinstaladas.

RAM síncrona

La memoria *RAM* dispone de una capacidad de 24 kbits y está organizada en forma de 6 k x 4 bits. El hardware está concebido para accesos de lectura y escritura de bytes, words o dwords. Es compatible con transferencias en modo ráfaga o "burst". Más información sobre la zona de direcciones de la memoria RAM: véase Apartado 5.

La memoria RAM integrada también puede describirse y leerse con ayuda del programa de monitorización suministrado. Información detallada: Véase Apartado 8 '*PPLABMON.EXE*'.

FIFO ("primero en entrar, primero en salir")

En la actualidad, la aplicación de ejemplo *Memoria FIFO (PEPS)* aún no está incluida en el volumen de suministro de la *PCIe-BaseLab*. En breve será puesta a disposición de nuestros clientes.

Registro de entradas/salidas de 16 bits

Esta aplicación de ejemplo se compone de un *Registro de datos de 16 bits* para la emisión de datos estadísticos (a través de circuito latch), así como una ruta de datos de entrada por separado a través de la que se puede hacer una consulta acerca del estado lógico momentáneo de 16 conducciones de entrada. En total, la aplicación de ejemplo "Registro de entrada/salida de 16 bits" utiliza 32 pines de entre las entradas/salidas disponibles en las regletas de patas (*pin header block*): 16 pines de entrada y 16 pines de salida. Están dispuestos en las regletas de tal manera que es posible puentearlos fácilmente bit por bit mediante una clavija de cortocircuito. De este modo, es posible volver a leer la palabra de datos emitida a través del circuito latch de salida.

El circuito latch de entrada/salida de 16 bits puede alcanzarse a través de una dirección de acceso (escribir y leer) existente en la zona de direcciones de memoria de la tarjeta (véase Apartado 5).

El registro de entrada/salida de 16 bits también puede describirse y leerse con ayuda del programa de monitorización suministrado (véase Apartado 8 '*PPLABMON.EXE*').

El lector encontrará en el Anexo la situación que presentan en el bloque los pines de entrada/salida del registro de 16 bits, expresada en forma de tabla.

5. La interfaz de programación

La interfaz de programación posibilita una serie de funciones que pueden utilizarse a partir de los programas C/C++ a fin de establecer una comunicación con la *PCIe-BaseLab* tarjeta. Para utilizar esas funciones, se dispone de los siguientes archivos:

- PPLABAPI.H: archivo de encabezado C – definición de las funciones de API (interfaz de programación de aplicaciones)
- PPLABERR.H: archivo de encabezado C – definición de las constantes de error,
- PPLABAPI.DLL: biblioteca de vínculos dinámicos – Implementación de las funciones de API (interfaz de programación de aplicaciones)
- PPLABAPI.LIB: biblioteca de importación para la carga implícita del archivo DLL de las aplicaciones de Visual C++.

A fin de poder utilizar las API-funciones, deben cargarse el archivo de encabezado y el driver API-DLL.

Utilización de los archivos

En los C/C++ archivos fuente hay que cargar el archivo con un comando, a fin de poder acceder a las funciones.

El archivo puede 'PPLABERR.H' utilizarse para la búsqueda del código de error, y habitualmente no debe ser incluido ("included").

Carga de los drivers-API-DLL

La carga de los trailers drivers-API-DLL puede producirse

- de manera explícita con ayuda de las funciones de Win32 "LoadLibrary" ("cargar biblioteca") y "GetProcAddress" ("obtener dirección de proceso") o bien
- de manera implícita utilizando la biblioteca de importación suministrada

El programa 'LoadDLL' de ejemplo demuestra la carga explícita del archivo DLL.

Al utilizar los entornos de desarrollo de Microsoft, la biblioteca de importación suministrada puede utilizarse para la carga implícita del archivo DLL. En todos los programas de ejemplo (salvo en "LoadDLL") se utiliza la carga implícita.

Formato de las funciones, código de error

Todas las funciones de la interfaz tienen un prototipo similar. El nombre de la función comienza siempre con el prefijo '`PPLAB_`'.

El valor devuelto por las funciones es siempre del tipo '`DWORD`', y señaliza si la función se ha realizado con éxito. El valor de devolución "0" indica que la función pudo ejecutarse sin errores. Un valor distinto de "0" señaliza un error y representa un código de error, que transmite información acerca de la causa del mismo. Los posibles códigos de error están definidos en el archivo 'PPLABERR.H' y utilizan valores entre `0x20010000..0x20FF0000`. Además, se añaden también a menudo códigos de error del sistema operativo, que utilizan los bits `0...11`.

Si una API-función suministra datos, éstos se preparan mediante parámetros de referencia (indicadores de dirección). De este modo, también es posible enviar varias informaciones al mismo tiempo.

Configuraciones de las memorias

La *PCIe-BaseLab* tarjeta viene equipada de serie con una memoria de 32 kbytes. Dicha memoria se utiliza para demostrar posibles aplicaciones. Dentro de estas aplicaciones, por ejemplo, se incluyen:

- Escribir y leer varios kbytes de datos
- Fijar y consultar pines de entrada/salida
- Iniciar transferencias continuadas de datos a través de acceso directo a la memoria (DMA)

Le rogamos que consulte la respectiva configuración de su(s) tarjeta(s) en la documentación adjunta.

Ejemplo de configuración 1:

```
0x0000..0x5FFF: RAM (24 kBytes)
0x6000..0x6001: I/O-REGISTER (16 Bits)
0x6002..0x7FFF: no
```

Métodos de acceso a la memoria

Hay tres métodos disponibles para poder acceder a la memoria de la *PCIe-BaseLab* tarjeta:

- KERNEL
- MAPPED
- DMA

El KERNEL método es el modo estándar de procedimiento para acceder a la zona de memoria de una PCI tarjeta externa inaceptable. Así, el acceso a la memoria se produce a través del WINDOWS-driver de la *PCIe-BaseLab* tarjeta al nivel del módulo central. El driver se ocupa también de la sincronización de accesos simultáneos a la memoria. A esto corresponden las funciones `PPLAB_ReadMemory` y `PPLAB_WriteMemory`.

Para el método están disponibles las funciones `PPLAB_MapMemory` y `PPLAB_UnmapMemory`. La correspondiente zona de memoria de la *PCIe-BaseLab* tarjeta se "intercala" en el espacio de direcciones de la aplicación en cuestión. A continuación, es posible realizar accesos directos a esta zona de memoria sin necesidad de "dar un rodeo" a través del nivel del módulo central. Para evitar accesos simultáneos, es necesaria una sincronización explícita. El programa *'MapMem'* de ejemplo ilustra este método.

El método sirve para transportar grandes volúmenes de datos mediante la utilización de los canales del controlador. Para ello se utilizan las funciones `PPLAB_ReadPlxRegister` y `PPLAB_WritePlxRegister`. El programa *'UseDMA'* de ejemplo muestra el modo de proceder necesario.

6. Funciones - Referencia

La interfaz de programación incluye las siguientes funciones de API (interfaz de programación de aplicaciones):

- Consulta del número de *PCIe-BaseLab* tarjetas activas
  `PPLAB_GetNumberOfDevices`
- Apertura y cierre de una *PCIe-BaseLab* tarjeta
  `PPLAB_OpenDevice`
  `PPLAB_CloseDevice`
- Consulta de la versión de driver
  `PPLAB_GetDriverVersion`
- Consulta de informaciones sobre la ranura de extensión de una *PCIe-BaseLab* tarjeta
  `PPLAB_GetDeviceProperties`
- Reinicio de una *PCIe-BaseLab* tarjeta
  `PPLAB_ResetDevice`
- Lectura y escritura de los registros existentes en la zona de configuración de una *PCIe-BaseLab* tarjeta
  `PPLAB_ReadPciRegister`
  `PPLAB_WritePciRegister`
- Lectura y escritura de los registros del controlador de una *PCIe-BaseLab* tarjeta
  `PPLAB_ReadPlxRegister`
  `PPLAB_WritePlxRegister`
- Lectura y escritura del contenido de una *PCIe-BaseLab* tarjeta
  `PPLAB_ReadEEPROM`
  `PPLAB_WriteEEPROM`
- Lectura y escritura en la zona de memoria de una *PCIe-BaseLab* tarjeta
  `PPLAB_ReadMemory`
  `PPLAB_WriteMemory`
- "Intercalación" y "enmascaramiento" de una zona de memoria de una *PCIe-BaseLab* tarjeta
  `PPLAB_MapMemory`
  `PPLAB_UnmapMemory`
- Consulta de la dirección física de las zonas de memoria de una *PCIe-BaseLab* tarjeta
  `PPLAB_GetPhysicalAddressOfUserRegion`
- Adjudicación y activación de zonas de memoria relacionadas
  `PPLAB_AllocateContMemory`
  `PPLAB_FreeContMemory`

Consulta del número de *PCIe-BaseLab*-tarjetas activas

```
DWORD PPLAB_GetNumberOfDevices (DWORD* pNumberOfDevices);
```

Mediante esta función es posible consultar cuántas *PCIe-BaseLab* tarjetas están activas actualmente en el ordenador. Para ello, es necesario introducir la dirección de una `DWORD` variable como parámetro `pNumberOfDevices`. Tras la ejecución exitosa de la función, esta variable recibe el número de tarjetas activas.

El *PCIe-BaseLab* driver es compatible con un máximo de 16 *PCIe-BaseLab* tarjetas por cada PC.

Habitualmente, esta función es la primera a la que se accede al iniciar un programa, a fin de determinar si existe alguna *PCIe-BaseLab* tarjeta activa en el ordenador.

Ejemplo:

```
DWORD error, deviceNumber;
error = PPLAB_GetNumberOfDevices(&deviceNumber);
if (error)
… // error handling
else
printf ("Active BaseLab-Cards found: %d\n", deviceNumber);
```

Apertura de una *PCIe-BaseLab t*arjeta

```
DWORD PPLAB_OpenDevice (HANDLE* pHDevice, DWORD deviceId);
```

Mediante esta función se inicia la utilización de una tarjeta: se abre el "aparato" con el parámetro `deviceId`. Como `deviceId` es posible introducir un valor entre `1` y `16`. Si sólo hay una tarjeta activa, ésta tiene la Id `1`, si hay dos tarjetas se llaman siempre `1` y `2`, etc. En caso de realizarse con éxito, la función devuelve un indicador o "handle" para el aparato correspondiente. Para ello, es necesario introducir la dirección de una variable como parámetro.

Dado que el `deviceHandle` será necesario como parámetro en todas las demás API-funciones (a excepción de `PPLAB_GetNumberOfDevices`), lo primero que se abra mediante esta función debe ser siempre una tarjeta, para que a continuación sea posible acceder a otras funciones.

Una *PCIe-BaseLab* tarjeta abierta debe volver a cerrarse mediante la función `PPLAB_CloseDevice` antes de que se finalice el programa en cuestión.

Ejemplo:

```
DWORD error, deviceId;
HANDLE deviceHandle;
deviceId = 1;


error = PPLAB_OpenDevive (&deviceHandle, deviceId);
if (error)
… // error handling
else
printf ("BaseLab-Card %d opened! (handle=%d)\n", deviceId, deviceHandle);
```

Cierre de una *PCIe-BaseLab* tarjeta

```
DWORD PPLAB_CloseDevice (HANDLE hDevice);
```

Mediante esta función se vuelve a cerrar una *PCIe-BaseLab*-tarjeta. Como parámetro hay que introducir el `deviceHandle` que fue creado con `PPLAB_OpenDevice`. De este modo, el `deviceHandle` deja de ser válido. Así, tras aplicar `PPLAB_CloseDevice`, ya no puede accederse a ninguna otra API-función para esta tarjeta.

Ejemplo:

```
…
error = PPLAB_CloseDriver (deviceHandle);
if (error)
… // error handling
```

Consulta de la versión del driver de Windows

```
DWORD PPLAB_GetDriverVersion (HANDLE hDevice, DWORD* pDriverVersion);
```

Esa función sirve para la consulta de la versión del driver de Windows instalado. Además del `deviceHandle`, debe introducirse como parámetro la dirección de una variable, que después incluirá el número de versión del driver Windows.

Ejemplo:

```
…
DWORD driverVersion;
error = PPLAB_GetDriverVersion (deviceHandle, &driverVersion);
if (error)
… // error handling
else
printf ("Current driver version: %x\n", driverVersion);
```

Consulta de informaciones sobre la ranura de extensión

```
DWORD PPLAB_GetDeviceProperties (HANDLE hDevice, DWORD* pSlotNum, DWORD* pBusNum,
                                 DWORD* pDeviceNum, DWORD* pFunctionNum);
```

Mediante esta función, es posible consultar informaciones sobre la ranura de extensión de la *PCIe-BaseLab* tarjeta en cuestión. Para ello, hay que introducir las direcciones de cuatro `DWORD` variables, que tras la ejecución de la función reproducirán los valores de los números de ranura, bus, aparato y función. Mediante estas informaciones es posible realizar una identificación en caso de que haya varias *PCIe-BaseLab* tarjetas activas.

Ejemplo:

```
…
DWORD slotNum, busNum, devNum, funNum;
error = PPLAB_GetDeviceProperties (deviceHandle, &slotNum, &busNum, &devNum,
&funNum);
if (error)
… // error handling
else
printf ("Device properties: slot=%d, bus=%d, dev=%d, fun=%d\n",
         slotNum, busNum, devNum, funNum);
```

Reinicio de una *PCIe-BaseLab* tarjeta

```
DWORD PPLAB_ResetDevice (HANDLE hDevice, DWORD resetFlags);
```

Al aplicar esta función, se lanza un reinicio de la *PCIe-BaseLab* tarjeta en cuestión. Hasta ahora no se ha definido ningún valor para el `resetFlags` parámetro, por lo que debe introducirse el valor `0`.

Ejemplo:

```
error = PPLAB_ResetDevice (deviceHandle, 0);
if (error)
… // error handling
else
printf ("Device reset done!\n");
```

Lectura de los registros existentes en la zona de PCI-configuración

```
DWORD PPLAB_ReadPciRegister (HANDLE hDevice, DWORD offset, DWORD* pValue);
```

Mediante esta función es posible hacer una lectura de una zona de PCI-configuración de la *PCIe-BaseLab* tarjeta, de 256 bytes de espacio. Como parámetro `offset` debe indicarse el offset en bytes (0,4,8..252), para el valor debe introducirse la dirección de una `DWORD` variable a modo de `pValue`.

Ejemplo:

```
DWORD value;
DWORD offset = 0;
error = PPLAB_ ReadPciRegister (deviceHandle, offset, &value);
if (error)
… // error handling
else
printf ("Vendor_Device_ID: 0x%X", value);
```

Escritura de los registros existentes en la zona de configuración PCI

```
DWORD PPLAB_WritePciRegister (HANDLE hDevice, DWORD offset, DWORD value);
```

Esta función sirve para describir los registros existentes en la zona de configuración PCI. Como parámetro `offset` debe indicarse el offset en bytes (0,4,8..252), y el valor a modo de parámetro `value`.

Por favor, tenga en cuenta: sólo se pueden describir algunos de los registros existentes en la zona de configuración PCI. Asigne a estos registros exclusivamente valores razonables.

Ejemplo:

```
DWORD subSystemId_subVendorId = 0x12345678;
DWORD offset = 0x2c;
error = PPLAB_ WritePciRegister (deviceHandle, offset subSystemId_subVendorId);
if (error)
… // error handling
```

Lectura de los registros del *PEX8311*-controlador

```
DWORD PPLAB_ReadPlxRegister (HANDLE hDevice, DWORD offset, DWORD* pValue);
```

Mediante esta función es posible hacer una lectura de los registros del PEX8311 controlador. Dentro de estos registros se incluyen el registro de configuración local, así como el registro de tiempo de activamiento (véase documentación sobre el PEX8311 controlador). Como parámetro `offset` debe indicarse el offset en bytes, para el valor debe introducirse la dirección de una `DWORD` variable a modo de `pValue`.

Ejemplo:

```
DWORD intCSR;
DWORD offset = 0x68;
error = PPLAB_ ReadPlxRegister (deviceHandle, offset, &intCSR);
if (error)
… // error handling
else
printf ("Interrupt_Control_Status: 0x%X", intCSR);
```

Escritura de los registros del *PEX8311*-controlador

```
DWORD PPLAB_WritePlxRegister (HANDLE hDevice, DWORD offset, DWORD value);
```

Esta función sirve para describir los registros del PEX8311 controlador. Como parámetro `offset` debe indicarse el adress offset en bytes, y el valor a modo de parámetro `value`.

Por favor, procure asignar exclusivamente valores razonables a los PEX8311 registros correspondientes.

Ejemplo:

```
DWORD p2lDoorbell = 0x00000001;
DWORD offset = 0x60;
error = PPLAB_ WritePlxRegister (deviceHandle, offset, p2lDoorbell);
if (error)
… // error handling
```

Lectura de datos de la EEPROM

```
DWORD PPLAB_ReadEeprom (HANDLE hDevice, DWORD offset, DWORD* pValue);
```

Mediante esta función es posible hacer una lectura de datos de la EEPROM de la *PCIe-BaseLab* tarjeta. Como parámetro `offset` debe indicarse el adress offset en bytes (0,4,8..0x60), para el valor debe introducirse la dirección de una `DWORD` variable a modo de `pValue`.

Ejemplo:

```
DWORD lat_gnt_pin_line;
DWORD offset = 0x0B;
error = PPLAB_ ReadEeprom (deviceHandle, offset, &lat_gnt_pin_line);
if (error)
… // error handling
else
printf ("MaxLat_MinGnt_IntPin_IntLine: 0x%X", lat_gnt_pin_line);
```

Escritura de datos en la EEPROM

```
DWORD PPLAB_WriteEeprom (HANDLE hDevice, DWORD offset, DWORD value);
```

Esta función sirve para la escritura de datos en la EEPROM. Como parámetro `offset` debe indicarse el offset en bytes (0,4,8..0x60), y el valor a modo de parámetro `value`.

IMPORTANTE - Por favor, tenga en cuenta:

Asegure el contenido original contra la modificación de EEPROM datos, por ejemplo a través del programa de monitorización PPLABMON. Asigne a los EEPROM registros exclusivamente valores razonables. Los contenidos erróneos pueden provocar errores de la *PCIe-BaseLab* tarjeta y del conjunto del sistema.

Ejemplo:

```
DWORD lat_gnt_pin_line = 0x00000100;
DWORD offset = 0x0B;
error = PPLAB_ WriteEeprom (deviceHandle, offset, lat_gnt_pin_line);
if (error)
… // error handling
```

Lectura de la zona de memoria

```
DWORD PPLAB_ReadMemory (HANDLE hDevice, DWORD offset, DWORD* pValue);
```

Mediante esta función es posible hacer una lectura de las memorias de la *PCIe-BaseLab* tarjeta. Como parámetro `offset` debe indicarse el offset en bytes (0,4,8..), para el valor debe introducirse la dirección de una `DWORD` variable a modo de `pValue`.

Ejemplo:

```
DWORD ioRegister;
DWORD offset = 0x6000;
error = PPLAB_ ReadMemory (deviceHandle, offset, &ioRegister);
if (error)
… // error handling
else
printf ("IORegister: 0x%X", ioRegister);
```

Escritura de la zona de memoria

```
DWORD PPLAB_WriteMemory (HANDLE hDevice, DWORD offset, DWORD value);
```

Esta función sirve para describir la *PCIe-BaseLab* memoria. Como parámetro `offset` debe indicarse el offset en bytes (0,4,8..), y el valor a modo de parámetro `value`.

Para obtener más información, consultar el Apartado “Configuraciones de las memorias”.

Ejemplo:

```
DWORD ioRegister = 0x0000FFFF;
DWORD offset = 0x6000;
error = PPLAB_ WriteMemory (deviceHandle, offset, ioRegister);
if (error)
… // error handling
```

"Intercalación" de una zona de memoria en el espacio de direcciones de la aplicación

```
DWORD PPLAB_MapMemory (HANDLE hDevice, USER_REGION userRegion, DWORD*
pVirtAddress);
```

Mediante esta función, se "intercala" una zona de memoria de la *PCIe-BaseLab* tarjeta en el espacio de direcciones de la aplicación. Con el parámetro `userRegion` se selecciona `USER_REGION_0` o `USER_REGION_1`. Como resultado de la función, una `DWORD` variable cuya dirección debe introducirse como parámetro `pVirtAddress` recibe la dirección virtual de esta zona de memoria. A continuación, a través de esta dirección es posible desde la propia aplicación escribir en la zona de memoria de la *PCIe-BaseLab* tarjeta o bien hacer una lectura de ella.

ATENCIÓN: En la configuración actual de la *PCIe-BaseLab* tarjeta sólo existe una zona de memoria. Por consiguiente, debe indicarse siempre como parámetro `USER_REGION_0`.

Ejemplo:

```
DWORD virtAddress;
error = PPLAB_ MapMemory (deviceHandle, USER_REGION_0, &virtAddress);
if (error)
… // error handling
else
{
DWORD* pMem = (DWORD*)(virtAddress + 4);        // set address to offset 4
value = *pMem;                                  // direct read DWORDs
pMem++;                                         // set address to offset 8
*pMem = 0x12345678;                             // direct write DWORDs
}
```

"Enmascaramiento" de una zona de memoria fuera del espacio de direcciones de la aplicación

```
DWORD PPLAB_UnmapMemory (HANDLE hDevice, USER_REGION userRegion, DWORD
virtAddress);
```

Mediante esta función, se vuelve a anular la "intercalación" de una zona de memoria con `PPLAB_MapMemory`. Como parámetro debe volver a introducirse el `userRegion` correspondiente, así como la dirección virtual que suministró `PPLAB_MapMemory`.

Tras aplicar esta función ya no es posible acceder directamente a esta zona de memoria.

Ejemplo:

```
error = PPLAB_ UnmapMemory (deviceHandle, USER_REGION_0, virtAddress);
if (error)
… // error handling
```

Consulta de la dirección física de las "Regiones de usuario"

```
DWORD PPLAB_GetPhysicalAddressOfUserRegion (HANDLE hDevice, USER_REGION userRegion,
                                            DWORD* pPhysAddress);
```

Para poder utilizar los DMA canales de la *PCIe-BaseLab* tarjeta, las direcciones físicas de las zonas de memoria implicadas deben estar disponibles.

Mediante esta función, es posible consultar la dirección física de la respectiva "Región de usuario". Con el parámetro `userRegion` se selecciona `USER_REGION_0` o `USER_REGION_1`. Como resultado de la función, una variable cuya dirección debe introducirse como `DWORD` parámetro recibe la dirección física de esta zona de memoria. A continuación, esta dirección puede utilizarse como dirección "local" al iniciar una DMA transferencia.

ATENCIÓN: En la configuración actual de la *PCIe-BaseLab* tarjeta sólo existe una zona de memoria. Por consiguiente, debe indicarse `USER_REGION_0` siempre como parámetro.

Ejemplo:

```
DWORD physLocalAddr;
error = PPLAB_GetPhysicalAddressOfUserRegion (hDevice, USER_REGION_0,
&physLpcalAddr);
if (error)
… // error handling
else
printf ("psysical address of region0: 0x%08X \n", physLocalAddr);
```

Adjudicación de una zona de memoria relacionada

```
DWORD PPLAB_AllocateContMemory (HANDLE hDevice, DWORD size, DWORD* pVirtAddress,
                                DWORD* pPhysAddress);
```

Cuando se está realizando una DMA transferencia es ventajoso poder utilizar una zona de memoria que había sido adjudicada físicamente en relación. Por tanto, es necesario averiguar las direcciones físicas de todas las páginas de memoria implicadas.

Mediante esta función, se solicita una zona de memoria físicamente relacionada. Como parámetro `size` debe indicarse el tamaño de la memoria en bytes. Para ello, sólo se consideran múltiplos de 4096 (tamaño de una página de la memoria). Como resultado de la función, la `DWORD` variable cuya dirección debe introducirse como parámetro `pVirtAddress` recibe la dirección virtual de esta memoria. A continuación, a través esta dirección es posible desde la propia aplicación escribir en esta zona de memoria de la tarjeta o bien hacer una lectura de ella. La variable cuya dirección se introduce como parámetro recibe la dirección física de esta zona de memoria. A continuación, esta dirección puede utilizarse como `PCI` dirección al iniciar una DMA transferencia.

El tamaño de la zona de memoria no debe sobrepasar un valor de 0x7FFFFF (dezimal: 8388607, 8MB – 1), ya que éste es el número máximo de bytes posible para una DMA transferencia. (Bytes 0..22 DMASIZx registros)

En la versión actual del driver de Windows pueden solicitarse hasta 32 zonas de memoria relacionadas con estas características.

Ejemplo:

```
DWORD memSize, virtPCIAddr, physPCIAddr;;
memSize = 1024*1024*2;            // request off 2 MBytes
error = PPLAB_AllocateContMemory( hDevice, memSize, &virtPCIAddr,
&physPCIAddr );
if (error)
… // error handling
else
printf ("virtual: 0x%08X, psysical: 0x%08X \n", virtPCIAddr, physPCIAddr);
```

Activación de la zona de memoria relacionada

```
DWORD PPLAB_FreeContMemory (HANDLE hDevice, DWORD virtAddress);
```

Mediante esta función, la zona de memoria que ha sido solicitada a través de `PPLAB_AllocateContMemory` vuelve a activarse. Como parámetro `virtAddress` hay que introducir la dirección virtual.

Ejemplo:

```
error = PPLAB_FreeContMemory (hDevice, virtAddr);
if (error)
… // error handling
```

7. Los programas de ejemplo (C++)

Los programas de ejemplo muestran el modo de proceder necesario para la utilización de las distintas API funciones de la *PCI-BaseLab* interfaz de programación. Se trata de aplicaciones de consola cuya programación se ha mantenido intencionadamente sencilla. A cada programa de ejemplo le corresponde un archivo con el mismo nombre. La programación debe ser posible con cualquier compilador. En el caso de herramientas de Microsoft es posible utilizar la biblioteca de importación.

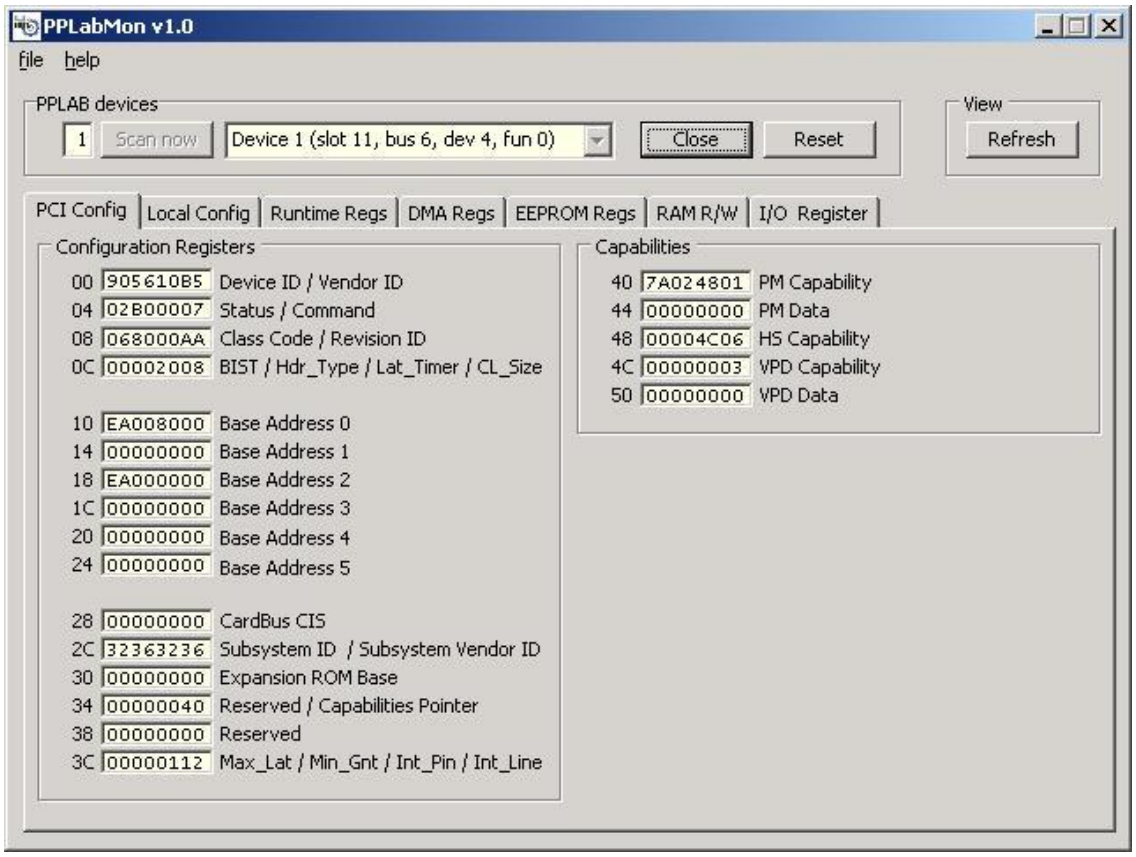
En los directorios se encuentran, además del archivo, los archivos de proyecto utilizados para la elaboración de los programas. (MS Visual C++ 6.0)

Se dispone de los ejemplos siguientes:

- APITest - aplicación de las funciones básicas de la interfaz de programación de aplicaciones (API)
- LoadDLL - carga explícita de los drivers-API-DLL
- MapMem - lectura y escritura directas en la zona de memoria "intercalada"
- RdWrRAM - lectura y escritura en la memoria con el método de módulo central o "KERNEL"
- ReadDMARegs - lectura DMA registros
- ReadLocalCfg - lectura 'local config register'
- ReadPCICfg - lectura 'PCI config register'
- ReadRuntimeRegs - lectura 'runtime register'
- Reset - reinicio de la *PCI-BaseLab* tarjeta
- TestEEPROM - lectura y escritura EEPROM registros
- UseDMA - inicio y arranque de DMA-transferencias

8. El programa de monitorización PPLABMON.EXE

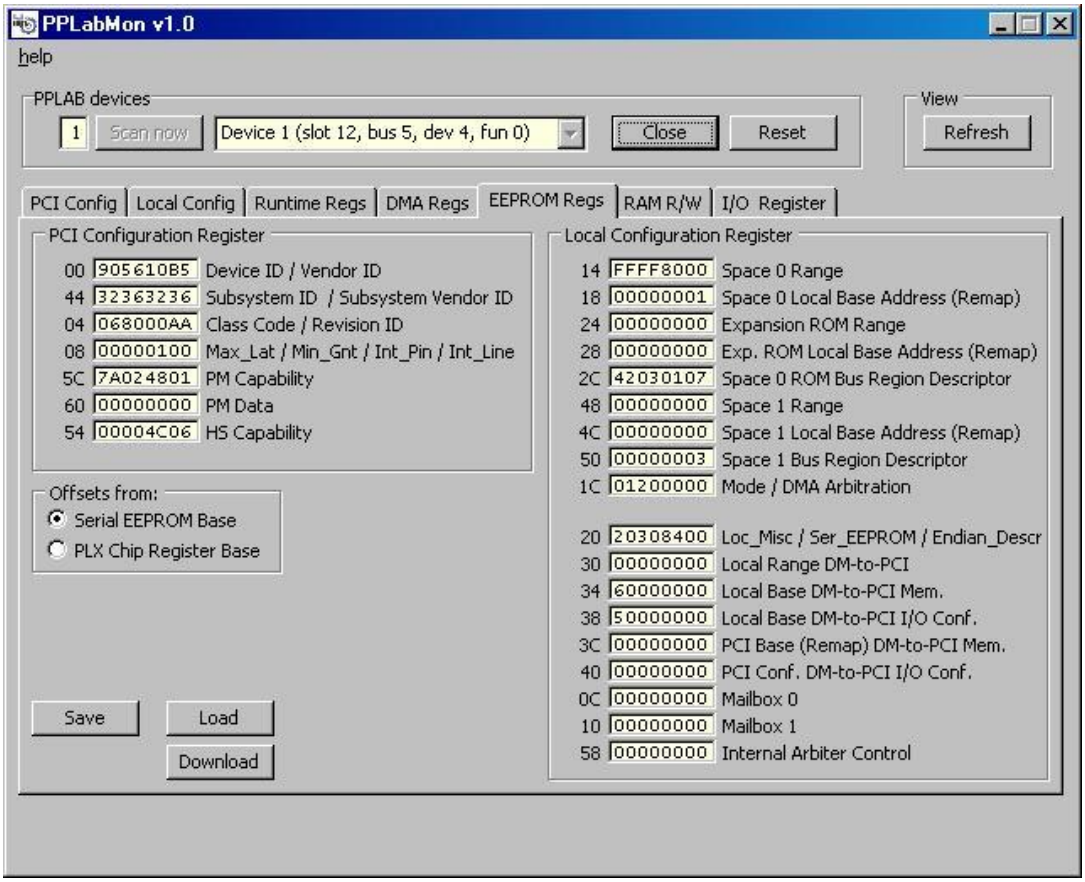
El programa de monitorización 'PPLABMON.EXE' es una aplicación gráfica de Windows que proporciona una vista exhaustiva de todas las zonas de la *PCIe-Baselab* tarjeta.



Están disponibles las siguientes posibilidades:

- PCI Config - lectura 'PCI config register'
- Local Config - lectura 'local config register'
- RuntimeRegs - lectura 'runtime register'
- DMARegs - lectura DMA registros
- EEPROMRegs - procesamiento de los registros
- RAM R/W - inicio y arranque RAM
- I/O Register - fijación y consulta de los pines del registro de entrada/salida de 16 bits

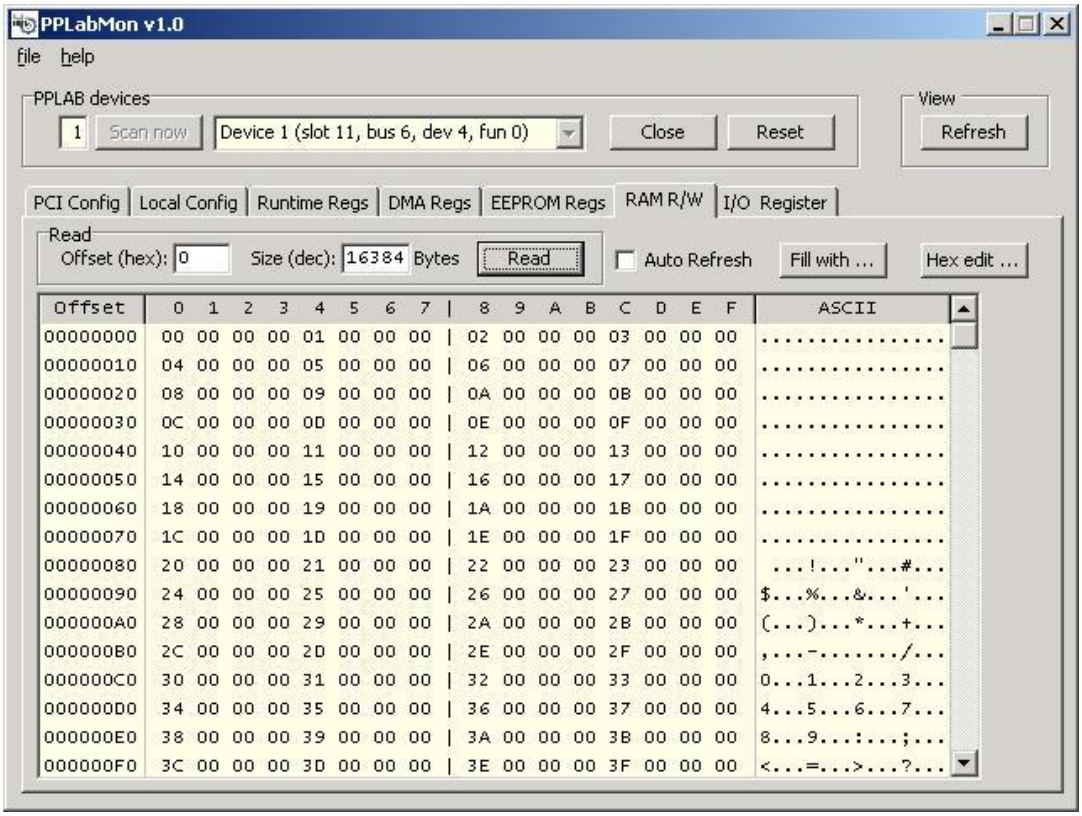
Procesamiento del registro de la EEPROM


PPLabMon v1.0
help
PPLAB devices
1
Scan now
Device 1 (slot 12, bus 5, dev 4, fun 0)
Close
Reset
View
Refresh
PCI Config
Local Config
Runtime Regs
DMA Regs
EEPROM Regs
RAM R/W
I/O Register
PCI Configuration Register
00 90561085 Device ID / Vendor ID
44 32363236 Subsystem ID / Subsystem Vendor ID
04 068000AA Class Code / Revision ID
08 00000100 Max_Lat / Min_Gnt / Int_Pin / Int_Line
5C 7A024801 PM Capability
60 00000000 PM Data
54 00004C06 HS Capability
Offsets from:
Serial EEPROM Base
PLX Chip Register Base
Save
Load
Download
Local Configuration Register
14 FFFF8000 Space 0 Range
18 00000001 Space 0 Local Base Address (Remap)
24 00000000 Expansion ROM Range
28 00000000 Exp. ROM Local Base Address (Remap)
2C 42030107 Space 0 ROM Bus Region Descriptor
48 00000000 Space 1 Range
4C 00000000 Space 1 Local Base Address (Remap)
50 00000003 Space 1 Bus Region Descriptor
1C 01200000 Mode / DMA Arbitration
20 20308400 Loc_Misc / Ser_EEPROM / Endian_Descr
30 00000000 Local Range DM-to-PCI
34 60000000 Local Base DM-to-PCI Mem.
38 50000000 Local Base DM-to-PCI I/O Conf.
3C 00000000 PCI Base (Remap) DM-to-PCI Mem.
40 00000000 PCI Conf. DM-to-PCI I/O Conf.
0C 00000000 Mailbox 0
10 00000000 Mailbox 1
58 00000000 Internal Arbiter Control

En la página "Registros de la EEPROM" están disponibles las siguientes posibilidades:

- leer registro - hacer una lectura del contenido actual de la EEPROM ('Refresh')
- modificar registro - modificar y "descargar" un registro de la EEPROM (doble clic)
- guardar en archivo - almacenar todos los registros en un archivo con formato editable ('Save')
- carga desde archivo - carga de todos los contenidos de registro desde un archivo ('Load')
- descarga general - descargar todos los registros en la EEPROM ('Download')

Lectura y escritura en la memoria RAM



En la 'RAM R/W' página están disponibles las siguientes posibilidades:

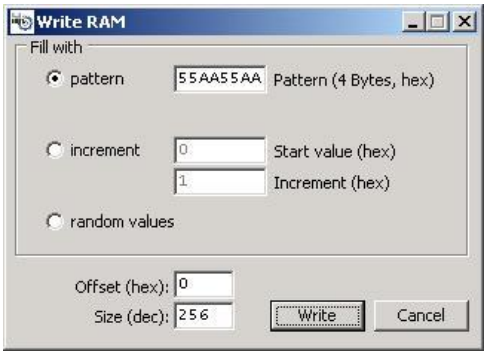
- modificar offset - determinar la dirección de inicio
- modificar RAM tamaño - ajustar el RAM tamaño mostrado
- lectura RAM - hacer una lectura del contenido actual ('Read')
- renovación automática - el RAM contenido se lee y se actualiza automáticamente
- escribir en la memoria RAM - llenar la memoria RAM con distintos valores ('Fill with...')
- modificación de bytes por separado - editar bytes por separado ('Hex edit...')

<u>Escribir</u>

Hay disponibles las siguientes posibilidades para escribir en la memoria RAM:

- RAM llenar con un patrón ('pattern')
- RAM llenar mediante incrementación ('increment')
- RAM llenar con valores aleatorios (random values')

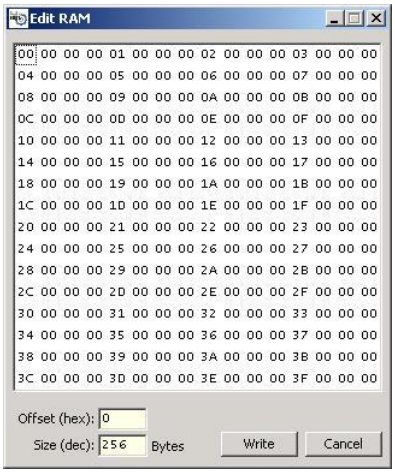
En esta ventana, todos los campos son editables.



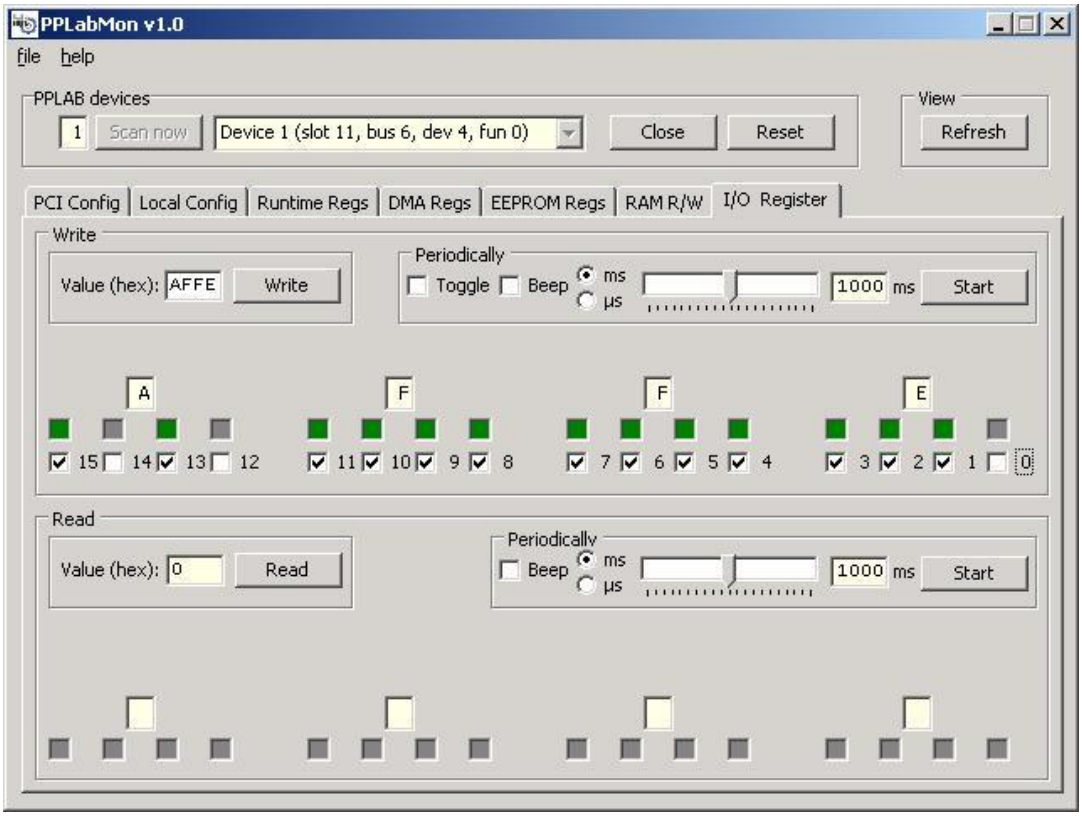
<u>Modificación de bytes por separado</u>

Esta ventana editable muestra 256 bytes, comenzando por el offset previamente ajustado. Para realizar la modificación, hay que seguir estos pasos:

- seleccionar el byte a evitar
- iniciar el modo: doble clic
- introducir el valor nuevo y confirmarlo mediante "Enter"
- mediante "Write", escribir el valor modificado en la memoria RAM



Fijación y consulta de los pines del registro

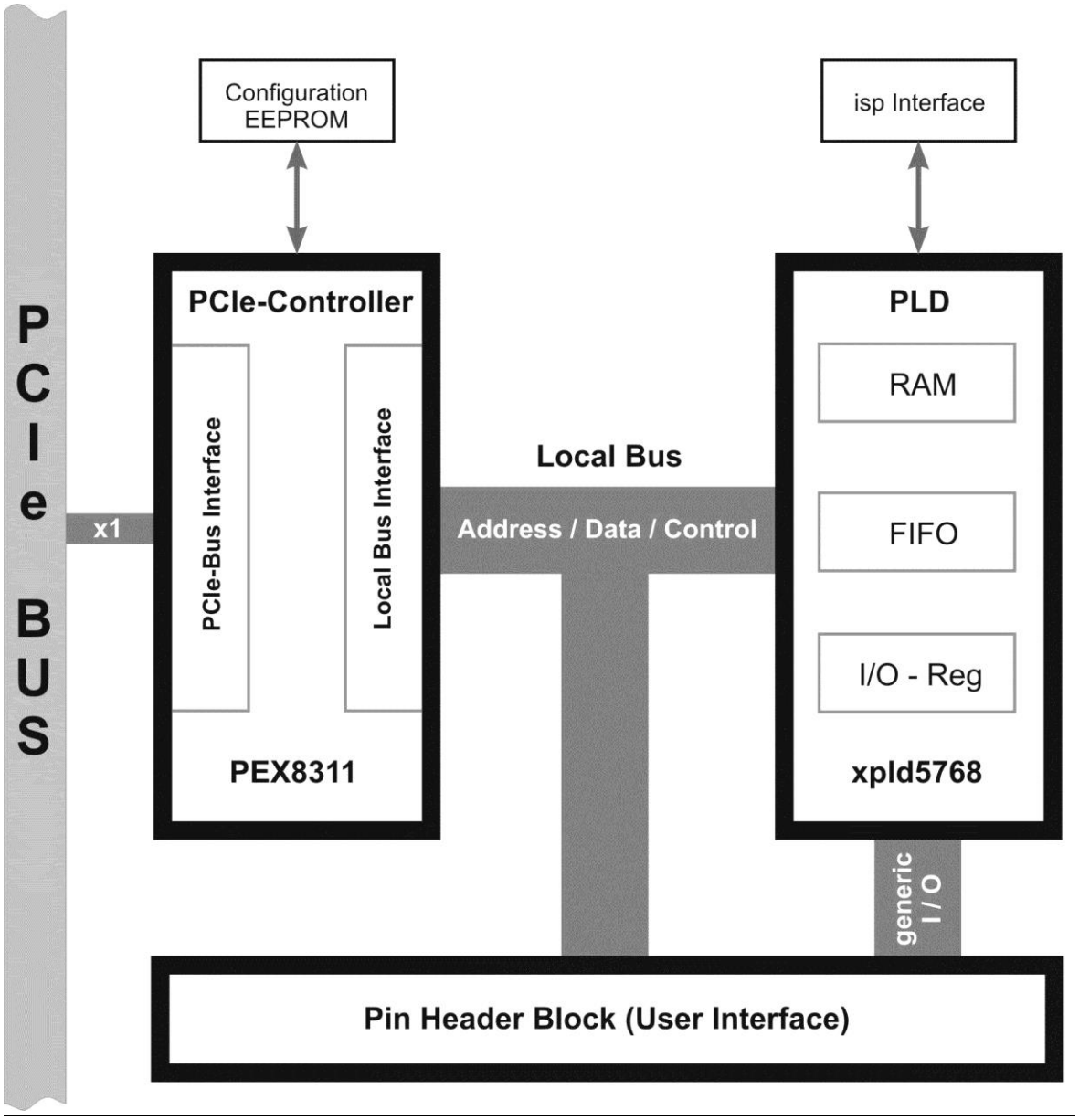


En la 'I/O Register' página están disponibles las siguientes posibilidades:

- fijación del 16-Bit-I/O-registro - Los pines pueden fijarse a modo de valor o bit a bit, y mediante "Write" se transmiten los ajustes al registro
- consulta del 16-Bit-I/O-registro - consulta de los pines
- escritura/lectura periódica - mediante el regulador es posible ajustar diferentes tiempos, "Start" activa la escritura/lectura periódica
- señal acústica ('Beep') - suena una señal acústica que avisa de la escritura/lectura
- negación bit a bit de los valores - cambio de los estado lógicos de los bits por separado ('Toggle')

Anexo

Diagrama funcional



Disposición de las conexiones

Pin	Name	Function	Pin	Name	Function
01	GND	GND	02	GND	GND
03	LCLK_JP	local clock output	04	GCLK1	pld clock 1 input
05	GCLK2	pld clock 2 input	06	GCLK3	pld clock 3 input
07	P30/D1	pld GPI/O	08	P28/E1	pld GPI/O
09	P26/F4	pld GPI/O	10	P24/F5	pld GPI/O
11	P22/E2	pld GPI/O	12	P20/CLK_OUT0/F2	pld GPI/O / pll0 CTL
13	P18/F1	pld GPI/O	14	P16/G1	pld GPI/O
15	P14/F3	pld GPI/O	16	P12/G5	pld GPI/O
17	P10/H5	pld GPI/O	18	P8/PLL_RST0/G4	pld GPI/O / pll0 CTL
19	P6/G3	pld GPI/O	20	P4/PLL_FBK0/H3	pld GPI/O / pll0 CTL
21	GND	GND	22	GND	GND
23	P2/G2	pld GPI/O	24	P0/H1	pld GPI/O
25	O30/C4	pld GPI/O	26	O28/E4	pld GPI/O
27	O26/B1	pld GPI/O	28	O24/C1	pld GPI/O
29	O22/D3	pld GPI/O	30	O20/C2	pld GPI/O
31	O18/E3	pld GPI/O	32	O16/D2	pld GPI/O
33	N5/A2	pld GPI/O	34	N4/B2	pld GPI/O
35	K30/D12	pld GPI/O	36	K28/B14	pld GPI/O
37	K26/C13	pld GPI/O	38	K24/A14	pld GPI/O
39	K22/A13	pld GPI/O	40	K21/B13	pld GPI/O
41	GND	GND	42	GND	GND
43	K20/D11	pld GPI/O	44	K18/B12	pld GPI/O
45	K16/C12	pld GPI/O	46	K14/E11	pld GPI/O
47	K4/E10	pld GPI/O	48	K2/A12	pld GPI/O
49	K0/A11	pld GPI/O	50	J14/C16	pld GPI/O
51	J12/B16	pld GPI/O	52	J10/C15	pld GPI/O
53	J8/B15	pld GPI/O	54	J6/E14	pld GPI/O
55	J4/D14	pld GPI/O	56	J2/E13	pld GPI/O
57	J0/A15	pld GPI/O	58	G8/M13	pld GPI/O
59	LD1/	PEX8311/Data	60	LD0	PEX8311/Data
61	GND	GND	62	GND	GND
63	LD3	PEX8311/Data	64	LD2	PEX8311/Data
65	LD5	PEX8311/Data	66	LD4	PEX8311/Data
67	LD7	PEX8311/Data	68	LD6	PEX8311/Data
69	LD9	PEX8311/Data	70	LD8	PEX8311/Data
71	LD11	PEX8311/Data	72	LD10	PEX8311/Data
73	LD13	PEX8311/Data	74	LD12	PEX8311/Data
75	LD15	PEX8311/Data	76	LD14	PEX8311/Data
77	LD17	PEX8311/Data	78	LD16	PEX8311/Data
79	LD19	PEX8311/Data	80	LD18	PEX8311/Data
81	GND	GND	82	GND	GND
83	LD21	PEX8311/Data	84	LD20	PEX8311/Data
85	LD23	PEX8311/Data	86	LD22	PEX8311/Data
87	LD25	PEX8311/Data	88	LD24	PEX8311/Data
89	LD27	PEX8311/Data	90	LD26	PEX8311/Data
91	LD29	PEX8311/Data	92	LD28	PEX8311/Data
93	LD31	PEX8311/Data	94	LD30	PEX8311/Data
95	DP2	PEX8311/Parity	96	DP3	PEX8311/Parity
97	DP0	PEX8311/Parity	98	DP1	PEX8311/Parity
99	GND	GND	100	GND	GND

Disposición de las conexiones por la parte de las patas

Pin	Name	Function	Pin	Name	Function
01	GND	GND	02	GND	GND
03	M30/B7	pld GPI/O	04	M28/A7	pld GPI/O
05	M26/D7	pld GPI/O	06	M24/C7	pld GPI/O
07	M22/B6	pld GPI/O	08	M21/E7	pld GPI/O
09	M20/E6	pld GPI/O	10	M18/A6	pld GPI/O
11	M12/B5	pld GPI/O	12	M14/A4	pld GPI/O
13	M8/B4	pld GPI/O	14	M10/A3	pld GPI/O
15	M5/C5	pld GPI/O	16	M6/B3	pld GPI/O
17	M2/D5	pld GPI/O	18	M4/C6	pld GPI/O
19	L30/B11	pld GPI/O	20	M0/D6	pld GPI/O
21	GND	GND	22	GND	GND
23	L26/B10	pld GPI/O	24	L28/C11	pld GPI/O
25	L22/C10	pld GPI/O	26	L24/A10	pld GPI/O
27	L20/C9	pld GPI/O	28	L21/D10	pld GPI/O
29	L12/A9	pld GPI/O	30	L18/E9	pld GPI/O
31	L8/E8	pld GPI/O	32	L14/F9	pld GPI/O
33	L5/B9	pld GPI/O	34	L10/F8	pld GPI/O
35	L2/B8	pld GPI/O	36	L6/A8	pld GPI/O
37	I30/H14	pld GPI/O	38	L4/D8	pld GPI/O
39	I26/G15	pld GPI/O	40	L0/C8	pld GPI/O
41	GND	GND	42	GND	GND
43	I22/PLL_RST1/H12	pld GPI/O / pll1 CTL	44	I28/G16	pld GPI/O
45	I18/F16	pld GPI/O	46	I24/PLL_FBK1/F15	pld GPI/O / pll1 CTL
47	I14/G13	pld GPI/O	48	I20/G14	pld GPI/O
49	I10/F14	pld GPI/O	50	I16/E16	pld GPI/O
51	M16/VREF0/A5	pld GPI/O / VREF0 Input	52	I12/G12	pld GPI/O
53	D10/VREF1/L9	pld GPI/O / VREF1 Input	54	I8/CLK_OUT1/E15	pld GPI/O / pll1 CTL
55	E20/VREV2/T14	pld GPI/O / VREF2 Input	56	EECS#	PEX8311/SPI_EPROM
57	L16/VREF3/D9	pld GPI/O / VREF3 Input	58	EECLK	PEX8311/SPI_EPROM
59	LRESET#	PEX8311/local rst output	60	EEWRDATA	PEX8311/SPI_EPROM
61	GND	GND	62	GND	GND
63	M_RST#	pld reset input	64	EERDDATA	PEX8311/SPI_EPROM
65	LHOLD	PEX8311/Control	66	DMPAF/EOT#	PEX8311/Control
67	LHOLDA#	PEX8311/Control	68	BIGEND#	PEX8311/Control
69	ADS#	PEX8311/Control	70	USERO/LLOCK0#	PEX8311/Control
71	BLAST#	PEX8311/Control	72	USERI/LLOCK1#	PEX8311/Control
73	READY#	PEX8311/Control	74	DREQ0#	PEX8311/Control
75	WAIT#	PEX8311/Control	76	DREQ1#	PEX8311/Control
77	LW/R#	PEX8311/Control	78	DACK0#	PEX8311/Control
79	BTERM#	PEX8311/Control	80	DACK1#	PEX8311/Control
81	GND	GND	82	GND	GND
83	CCS#	PEX8311/Control	84	MODE0	PEX8311/Control
85	LINTO#	PEX8311/Control	86	MODE1	PEX8311/Control
87	LINTI#	PEX8311/Control	88	I0/D15	pld GPI/O
89	LSERR#	PEX8311/Control	90	I2/D16	pld GPI/O
91	BREQI	PEX8311/Control	92	I4/F13	pld GPI/O
93	BREQO	PEX8311/Control	94	I6/F12	pld GPI/O
95	A6/J4	pld GPI/O	96	PMEIN#	PEX8311/Control
97	not connected	-	98	not connected	-
99	GND	GND	100	GND	GND

Disposición de las conexiones por la parte de las patas

Pin	Name	Function	Pin	Name	Function
01	GND	GND	02	GND	GND
03	LA31	PEX8311/Address	04	LA30	PEX8311/Address
05	LA29	PEX8311/Address	06	LA28	PEX8311/Address
07	LA27	PEX8311/Address	08	LA26	PEX8311/Address
09	LA25	PEX8311/Address	10	LA24	PEX8311/Address
11	LA23	PEX8311/Address	12	LA22	PEX8311/Address
13	LA21	PEX8311/Address	14	LA20	PEX8311/Address
15	LA19	PEX8311/Address	16	LA18	PEX8311/Address
17	LA17	PEX8311/Address	18	LA16	PEX8311/Address
19	LA15	PEX8311/Address	20	LA14	PEX8311/Address
21	LA13	PEX8311/Address	22	LA12	PEX8311/Address
23	LA11	PEX8311/Address	24	LA10	PEX8311/Address
25	LA9	PEX8311/Address	26	LA8	PEX8311/Address
27	LA7	PEX8311/Address	28	LA6	PEX8311/Address
29	LA5	PEX8311/Address	30	LA4	PEX8311/Address
31	LA3	PEX8311/Address	32	LA2	PEX8311/Address
33	GPIO0	PEX8311/GPI/O	34	LBE0#	PEX8311/Control
35	GPIO1	PEX8311/GPI/O	36	LBE1#	PEX8311/Control
37	GPIO2	PEX8311/GPI/O	38	LBE2#	PEX8311/Control
39	GPIO3	PEX8311/GPI/O	40	LBE3#	PEX8311/Control

Disposición de las conexiones por la parte de las patas

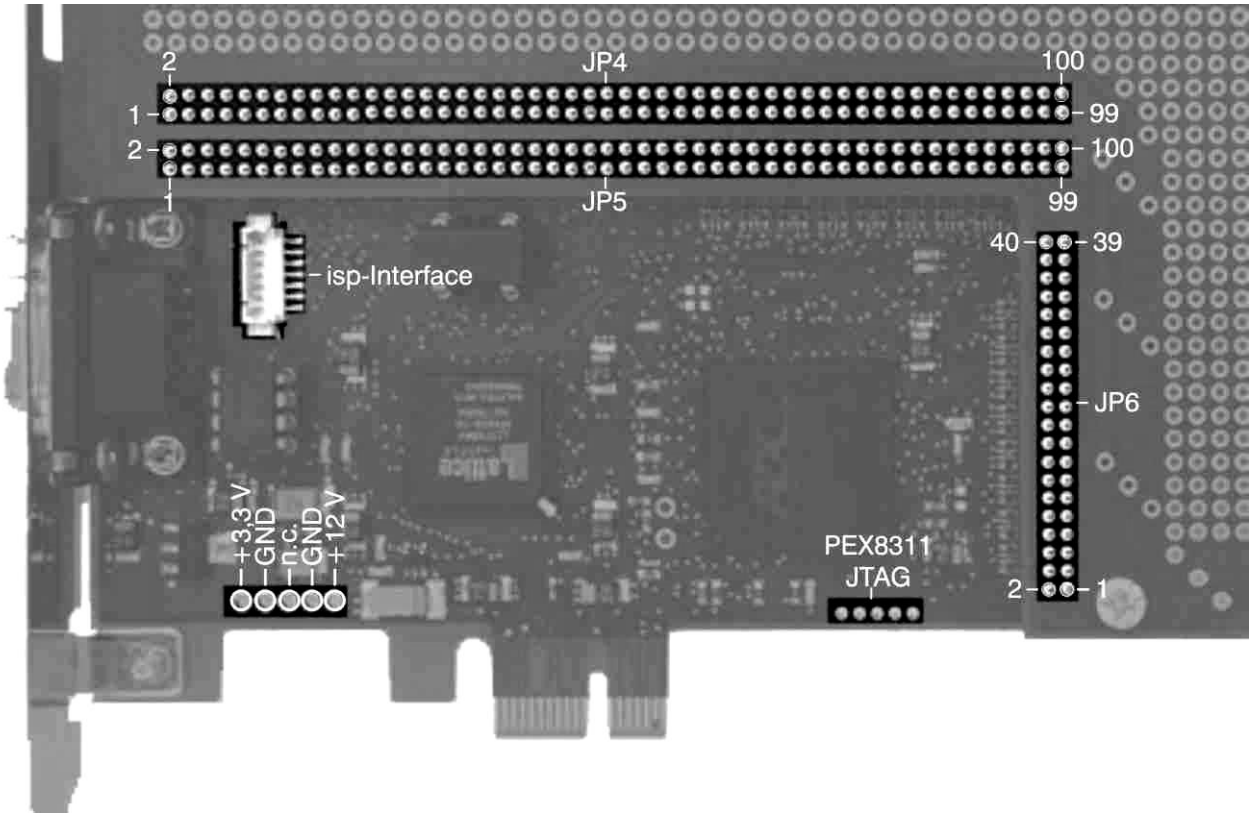
Disposición de las conexiones ISP-Interface (X1)

Pin	Name
01	GND
02	TDO
03	TDI
04	TMS
05	TCK
06	TOE
07	Not connected

Disposición de las JTAG conexiones PEX8311 (JP2)

Pin	Name
01	TDO
02	TRST#
03	TDI
04	TMS
05	TCK

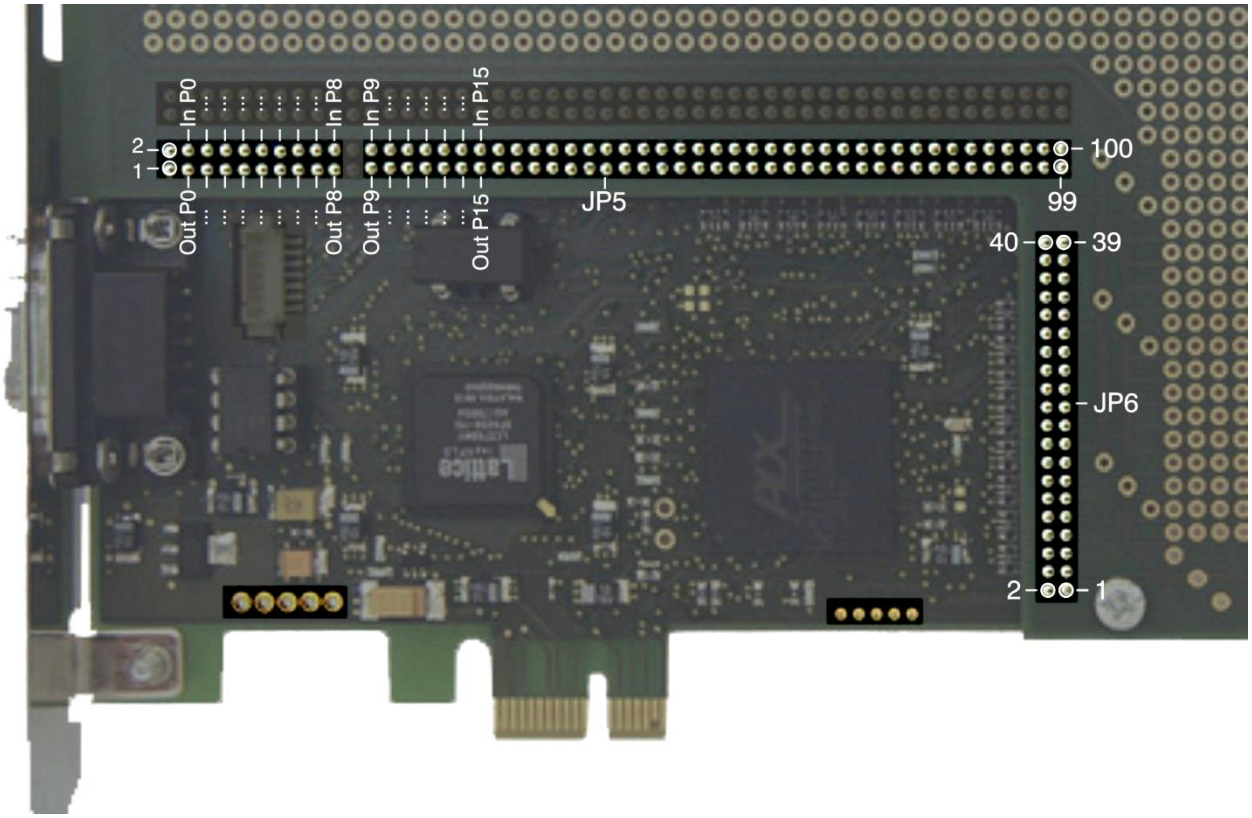
Ubicación de las regletas de patas / conectores



Aplicación de ejemplo ‘16Bit I/O-Register’ ubicación de los pines de conexión JP5

Pin	Name	Function	Pin	Name	Function
01	GND	GND	02	GND	GND
03	M30/B7	Out P0	04	M28/A7	In P0
05	M26/D7	Out P1	06	M24/C7	In P1
07	M22/B6	Out P2	08	M21/E7	In P2
09	M20/E6	Out P3	10	M18/A6	In P3
11	M12/B5	Out P4	12	M14/A4	In P4
13	M8/B4	Out P5	14	M10/A3	In P5
15	M5/C5	Out P6	16	M6/B3	In P6
17	M2/D5	Out P7	18	M4/C6	In P7
19	L30/B11	Out P8	20	M0/D6	In P8
21	GND	GND	22	GND	GND
23	L26/B10	Out P9	24	L28/C11	In P9
25	L22/C10	Out P10	26	L24/A10	In P10
27	L20/C9	Out P11	28	L21/D10	In P11
29	L12/A9	Out P12	30	L18/E9	In P12
31	L8/E8	Out P13	32	L14/F9	In P13
33	L5/B9	Out P14	34	L10/F8	In P14
35	L2/B8	Out P15	36	L6/A8	In P15

Disposición de las conexiones por la parte de las patas



Hotline:

HK Meßsysteme GmbH
Straße am Heizhaus 1
D-12557 Berlin /Germany

Telefon: ++49/30/633 75 114
Fax: ++49/30/633 75 116
E-Mail: support@pci-tools.de
Web: http://www.pci-tools.com
http://www.pci-tools.de

DriverFactory
Ostendstr. 25
D-10318 Berlin /Germany

Telefon: ++49/30/5304 2020
Fax: ++49/30/5304 2021
E-Mail: info@driverfactory.de
Web: http://www.driverfactory.de

PEX8311:

PLX Technology, Inc.
870 W. Maude Ave.
Sunnyvale, CA 94085
U.S.A.

Phone: ++1-408-774-9060
Fax: ++1-408-774-2169
Web: http://www.plxtech.com

xpld5768:

Lattice Semiconductor Corporation
5555 N.E. Moore Court
Hillsboro, Oregon 97124-6421
U.S.A.

Phone: ++1-503-268-8000
Fax: ++1-503-268-8347
Web: http://www.latticesemi.com

Webaddresses

http://www.pci-tools.com
http://www.pci-tools.de
http://www.driverfactory.de
http://www.plxtech.com
http://www.latticesemi.com
http ://www.pcisig.com